# Mobile Immersive Music

Jacques Lemordant, Agnes Guerraz

# MOBILE IMMERSIVE MUSIC

*J. Lemordant*

WAM project
jacques.lemordant
@inrialpes.fr

*A. Guerraz*

WAM project
agnes.guerraz
@inrialpes.fr

**ABSTRACT**

Due to obvious portability constraints, mobile technology excludes large electronic displays for visual immersion. On the contrary, sound heard over headphones is ideally suited for mobile applications. The use of stereo headphones or stereo speakers on mobile devices enables to take advantage of binaural technology which can provide an immersive sound experience for a variety of applications ranging from stereo widening of music (creating an out of the head listening experience) to full 3-D positional audio. Advances in audio are going to help bring in richer multimedia, increase quality of mobile music and help create more interactive and immersive audio applications. Interaction with sound in 3D audio space is no more limited to indoor environment [8]. In this paper, we report on an architecture for multimedia applications on mobile devices separating content creation (audio and graphics) from content manipulation. We have developed a markup format for interactive and spatialized audio on mobiles which can be used an interface between the sound designer and the application programmer. After presenting an overview of the key concepts in designing a format for interactive and spatialized audio and the methodology used to build the corresponding sound API , we describe its use in a mobile immersive music application for Copenhagen Channels where interactivity with the music is done through GPS waypoints.

## 1. INTRODUCTION

Most of mobiles are now shipped with a Java virtual machine and a java programming environment call Java Micro Edition (J2ME). Java virtual machines can be used to drive the interactive music at runtime, according to a set of rules established by the composer/editor and we can predict that more and more mobile audio applications will be adopting indeterminate adaptive digital music systems. An application programming interface (API) particularly interesting for mobile audio has been recently defined. This API is called Advanced Multimedia Supplements API or Java Specification Request 234 (JSR 234). JSR-234 is a well designed object oriented framework which enhances the audio support on mobile devices by adding rendering features like 3D audio, special audio effects and virtual acoustics. There is no specific support for interactive or indeterminate audio, no objects have been defined to manage time and synchronization at a higher level than that of the Java Mobile Media API or to manage the changing in time of DSP parameters. However object-oriented framework are easily extensible and in this paper, we will show how JSR-234 can be extended to provide a high level support to interactive audio.

## 2. MULTIMEDIA CONTENT DEVELOPMENT

Content development is the process of designing, writing, gathering, organizing, and editing content that may consist of graphics, audio, recordings, or other media assets that could be used on mobile devices. Content development is the responsibility of audio designers and visual designers and logic development the responsibility of software developers.

**Audio designer** - By audio designers we mean composers and/or sound designers, depending on the application and the content to be built. In the case of a game development, due to sound special effects, both are needed to achieve a complete audio rendering. Audio designers maintain direct control over the composition and presentation of an interactive sountrack. The audio designer recognizes the limitations of the medium and strives to engage interaction between the sound stimulus and the listener's interpretive ability.

**Visual designer** - The visual designer conceives, creates assets, and refines the look and feel for the application. This includes, but is not limited to the components of the application, user interface controls, iconography, and so forth. While visual designers may relay concepts through documents, mood boards, and other artifacts, their primary contribution to the development process is graphic assets.

**Software developer** - The software developer works with the live tree obtained by parsing the documents produced by audio and visual designers. He has to write the logic of the application, to detect collision between visual objects, to get input from the user or the system and to actualize the graph of audio and visual object inside the main loop at the right frequency.

## 3. INTERACTIVE AND SPATIALIZED AUDIO

A serie of documents regarding potential formats for interactive audio applications have been developed by groups such as the Interactive Audio special interest group (IAsig). In 1999, the Interactive Audio special interest group published an Interactive 3D Audio Rendering Guideline Level 2.0 (I3DL2). In 2005, the Synchronized Multimedia Activity of the World Wide Web Consortium (W3C) designed the Synchronized Multimedia Integration Language version 2.0 (SMIL 2.0) for choreographing multimedia presentations where audio, video, text and graphics are combined in real time. SMIL and I3DL2 can be joined as shown by Kari Pihkala and Tapio Lokki [4]. In their approach, visual and audio elements are defined in a same SMIL document.

This year, eight years after the completion of the I3DL2 guidelines, IASIG will announce the completion of a new interactive audio file format to complement I3DL2. This new format, based on the XMF file format, will be called Interactive XMF [3]. The goal of the IASIG in designing this format is to put artistic control into the hands of the artists, keep programmers from having to make artistic decisions, eliminate rework for porting to new platforms, and reduce production time, cost, and stress. The main objects in IXMF are cues which can be defined as a symbolic name associated to a graph of audio elements producing a continuous soundtrack from discrete media chunks. Clearly, iXMF with its 4-level hierarchical model for mixing and muting (track, chunk, cue, and mixgroups) is a complex file format, which has not been designed with mobiles in mind. The Synchronised Multimedia Integration Language (SMIL)[6] has been recommended by W3C and offers a way to synchronise text, video, graphics, audio and vector-based animation based on a timeline. SMIL-based timing and synchronization and SMIL-based declarative animation have not been used by the IASIG to define the iXMF format which is instead composed of C data structures and scripts.

We show in this paper that for mobile (and probably not only for mobile) a format for interactive and spatialized audio can be defined by extending the audio object-oriented format (close to I3DL2) specified in Java Advanced Multimedia Supplements (JSR 234). SMIL-based timing and synchronization and SMIL-based declarative animation for DSP parameters can be joined with the audio part of JSR-234 to get a declarative textual format with capabilities similar to iXMF but at a higher level of abstraction, introducing the concepts of 3D sound sources and audio special effects together with those of cues, layer, chunk and animation of DSP parameters.

## 4. AAML FORMAT AND API FOR MOBILES

We describe here the XML format that we have designed for Interactive and Spatialized Audio called Advanced Audio Markup Language (AAML). We need to define both the format and the corresponding API or framework. Hierarchical types systems with inheritance are the more powerful way to describe system of objects at a high level. Moreover, XML-Schema support inheritance and thus if we design a framework build around a hierarchical model with inheritance, it's possible to define an associated XML format by deriving the XML-Schema from it. The advantage of using an XML format is that we have many tools to transform it or even generate code from it. For example, it should be possible to start from AAML and generate java code for JSR-234 and C code for iXMF or well-known interactive audio API like FMOD [2]. Another advantage of using XML technology is that a graphical editor for music composers can be constructed more easily. A graphical editor allows to separate audio content creation from audio content manipulation as already explained. We are in the process of building such an editor for music composers.

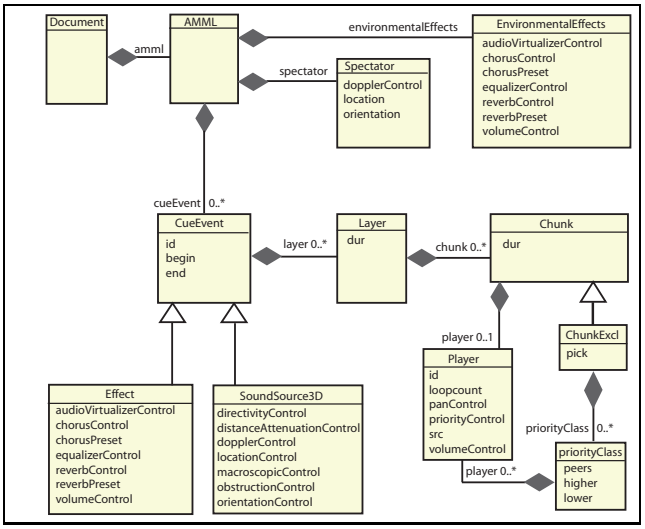


**Figure 1**. AAML XML-Schema

We decided to start from the API define by the JSR-234 for obvious reasons:

- JSR-234 is already implemented on new mobile phones such as the Nokia N95 and JSR-234 is a high level API.
- It will be easier to build a graphical authoring system allowing an audio-designer to produce AAML documents by embedding the JSR-234 engine in the authoring system.
- It will be easier for an application programmer to parse an AAML instance document and map it on JSR-234 API's objects.

A module or cueEvent is the main object of the JSR-234 API. The audio modules are of two types, SoundSources3D for 3D sounds and Effect for audioFX. A module aggregates players corresponding to audio chunks and with the JSR-234 GlobalManager, we have a 3-level hierarchical model. The position of the spectator or listener can be controlled in a 3D-space as for 3D sound sources.

We have extended the JSR-234 API with new types and called the extended API AAML.To add a high level support for time and synchronization, we introduce two new types, Layer and Chunk, and to bring support for indeterminate music we add the ChunkExcl (exclusive) type borrowing this element from SMIL 2.0. These types are shown on figure 1. With the help of timing attributes like begin, end, dur (duration), repeatCount, repeatDur, this brings in the AAML API a 5-level hierarchical model (Manager, CueEvent, Layer, Chunk, Player) for mixing, muting and rendering. CueEvent and Layer have a time semantic and are in fact synonym of parallel and sequential. Chunk in its Exclusive specialization brings selective or random composition. To summarise, we have three containers:

- *CueEvent* containers execute their children in parallel.
- *Layer* containers execute their *chunk* children in sequence.
- *ChunkExcl* containers execute one *chunk* child at a time as in a state machine.

Our model is therefore composed of cue events with layers playing in parallel, a layer being a group of chunks playing sequentially. Some chunks are containers of type ChunkExcl which then play their audio files in random order like an iPod in shuffle mode. To add support for animation of DSP parameters, we introduce three more new types, Track, KeyFrame and Controller. A Track associates a DSP property with a controller and keyFrame data. At the format level, these animation parameters are embeded in an *animate* element as shown on figure 3. These *animate* elements are borrowed form the SMIL 2.0 animation module. They are not shown on figure 1 to lower the complexity of the diagram.
The beginning and end of time and media elements may be linked to the execution of other elements, to user interface events or external events like a GPS track point reaching a way point. The ChunkExcl container is more than a state machine and offers a good support for indeterminate music: although only one child may play at a time, several other children may be queued. SMIL offers an attractive conceptual framework for doing so: priority classes. Each priority class is characterized by a peers attribute to determine how members of the class interrupt each other. The possible values are defer (don't interrupt, but put the interrupting peer into the queue), pause, never (don't start the interrupting peer), and stop (stop the current peer right away). A priority class may also determine specific policies for higher and lower priority elements.

## 5. IMMERSIVE MUSIC FOR COPENHAGEN CHANNEL

We choose to develop an innovative immersive music application on mobile phones. 3D spatialized sound sources positionned with the help of a geographical information


**Figure 2**. Copenhagen Channel Soundscape

system, play an immersive music composed using a graphical editor producing AMML documents. This application has some similarity with other existing applications like the Tactical Sound Garden developed by [5]. Our application will go further by placing musical sources and not only sounds in the landscape and by providing software based on the AMML format to help composers. The stabilization of the soundscape with respect to head mouvements is done through the use of a bluetooth sensor providing compass heading [7] [1]. This sensor is attached to the headphones and detected by the mobile phone if present . The Copenhagen Channel audio application allows its user to hear immersive music depending on her/his location while walking around the border of the Copenhagen channels on a predefined path. Music is coming from the 3D sound sources to the user depending on her/his spatial location. This application provides an immersive music experience inside a geographic area without having to deploy any audio devices on the site. Sound sources can be positionned anywhere and even underwater inside the channels. To do that, the mobile music composer has only to edit a local map on Google Maps to be able to place immersive sound spots as shown on Figure 2. The digital music files are embarqued in the external memory card of the phone. The 3D rendering of the soundsources in the proximity of the walker is done in realtime by the phone and the transition from one sound source to the other automatically made by the sound mixer of the phone.
If he desires it,the user/walker can have a visual feedback through the rendering of a mobile map (SVGT document) as shown on Figure 2. This map contains the layout of the channels, the sound source locations, and the user position. Figure 3, shows the AAML document which specifies the soundscape and the music played by the sound sources. Through this declarative approach, a change in the musical soundscape is easy: the composer has only to edit the text document. The spectator element of the AMML document instructs the system to use the GPS position of the device to get the spectator position. Heading is provided by the bluetooth compass if present, otherwise

```
<?xml version="1.0" encoding="UTF-8"?>
<p:amml xmlns:p="amml">
        <spectator location="fromGPS" />
        <environmentalEffects reverbPreset="underwater">
                <equalizerControl wetLevel="13" ModulationDepth="20"/>
        </environmentalEffects>

        <module xsi:type="SoundSource3D" id="SS1">
        <!-- google maps location + underwater location -->
                <locationControl location="12.602605,55.682022,-2.0"/>
                <cueEvent>
                        <layer>
                                <chunk xsi:type="Chunk" begin="0">
                                        <player src="takkelloftvej.mp3" loopCount="-1"/>
                                </chunk>
                        </layer>
                        <layer>
                                <chunk xsi:type="ChunkExcl" begin="0" pick="random">
                                        <priorityclass peers="pause">
                                                <player src="underwater1.mp3"/>
                                                <player src="underwater2.mp3"/>
                                                <player src="underwater3.mp3"/>
                                        </priorityclass>
                                </chunk>
                        </layer>
                </cueEvent>
        </module>

        <module xsi:type="Effect" id="FX1">
                <volumeControl level="24"/>
                <equalizerControl wetLevel="13" ModulationDepth=" 40"/>
                <reverbControl>
                        <animate attributeName="reverbTime" begin="0" end="13"
                                from="120" to="412" />
                </reverbControl>
                <cueEvent end="40s">
                        <layer>
                                <chunk xsi:type="Chunk" begin="0" dur="3s">
                                        <player src="intro1.mp3"/>
                                </chunk>
                                <chunk xsi:type="Chunk" begin="1" dur="indefinite">
                                        <player src="voice1.mp3"/>
                                </chunk>
                        </layer>
                </cueEvent>
        </module>
        ...
</p:amml>
```

**Figure 3**. ChannelSoundscape.aml document

from the GPS. The audio soundtrack is built in realtime from a mix of the 3D sound sources shown on Figure 2. Each 3D sound source will play, in an eventually random sequential order, chunks of immersive music. FX sound spots will help the user during his walk around the Copenhagen Channels.

This application is a J2ME application running on a Nokia N95 phone, the first phone on the market (2007) with a sophisticated sound mixer implementing the JSR 234 API. We have been using an emulator to develop and test the application with spatialized and interactive audio. The application (midlet) is downloadable from a web site (http://wam.inrialpes.fr/people/lemordant).

## 6. CONCLUSION

This work was motivated by the evidence gathered from having developing several 2D and 3D games for mobiles that a format and API for interactive and spatialized audio was missing. In this paper, we have presented an innovative XML format and its corresponding API to help creating interactive spatialized audio applications. This format is well adapted to mobile devices and allows us to separate content creation from content manipulation. A composer is less dependant on application programmers and reuse of audio cues in other applications is easier. In order to describe sound sources, special effects and environments properties, the Advanced Audio Markup Language (AAML) format uses an high level object-oriented approach which follow closely for 3D audio rendering the advanced multimedia API for mobiles described in the JSR 234. For interactive and indeterminate immersive audio aspects, time and synchronization, animation of digital signal processing (DSP) parameters, we have used multimedia SMIL based modules and embedded them in our markup language. Having a textual format using XML is a big advantage as it is easy to parse and generate code from the XML document. Starting from the AAML XML-schema, we are now building using the Eclipse-Graphical Modeling Framework an authoring system for interactive and spatialized audio, to put artistic control in the hand of a sound designer same as SVGT (M2G) or Mobile 3D graphics (M3G) allows to put artistic control in the hand of a visual designer using authoring system like Adobe illustrator or 3DS Max. JSR-234 API enhanced with the extensions for interactive and indeterminate music described in this paper is a very powerful system to build a wide range of mobile audio applications.

## 7. REFERENCES


[1] G. Essl and M. Rohs. The design space of sensing-based interaction for mobile music performance. *Proceedings of the 3rd International Workshop on Pervasive Mobile Interaction*, May 2007.

[2] FMOD. Fmod, a cross platform audio library. available on: `http://www.fmod.org/`.

[3] C. Grigg. 2001 annual interactive music conference, group report: Towards interactive xmf. available on: `http://www.projectbarbq.com`.

[4] K.Pihkala and T.Lokki. Extending smil with 3d audio. *Proceedings of the 2003 International Conference on Auditory Display*, July 2003.

[5] M. Shepard. Tactical sound garden toolkit. In *Mobile Music Workshop*, University of Sussex, Brighton, UK, March 2006.

[6] W3Consortium. The synchronized multimedia integration language: Smil 2.1. available on: `http://www.w3.org/AudioVideo/`.

[7] J. Williamson, R. murray Smith, and S. Hughes. Shoogle: Excitatory multimodal interaction on mobile devices. *CHI, International conference for human-computer interaction*, March 2007.

[8] M. Wozniewski, Z. Settel, and J. R. Cooperstock. A paradigm for physical interaction with sound in 3-d audio space. *ICMC*, 2006.